榨出iOS的所有潜力，让你的程序跑得更快！

Broadview
www.broadview.com.cn

iOS 性能优化实战

珲少 编著

·资深性·
一线资深工程师
5年的工作经验
与总结

·多图解·
用近200张图
详细讲解iOS优
化方法

·偏实战·
包含100多段
实用代码，领会
优化精髓

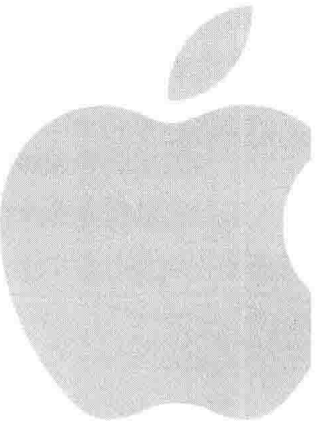
·很超值·
赠送720分钟
界面开发及优化
教学视频

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

iOS
性能优化实战

珲少 编著



電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书从 iOS 性能优化谈起，介绍了开发一款高性能的 iOS 应用程序所需要关注的技术点，并深入介绍了 iOS 高阶应用相关技巧。

全书主要包括如下几部分：iOS 应用内存管理的基本原理，以及内存管理的注意事项与检查内存问题的方法；iOS 应用的网络开发技能，以及网络调试与数据 Mock 技巧；iOS 应用程序的启动流程，以及推送与网络电话服务；iOS 视图渲染性能优化与动画技巧；完整的 iOS 多线程高级应用技术；Objective-C 语言的动态特性与 iOS 开发中运行时特性的应用，以及 JavaScript 脚本在 iOS 开发中的应用。

本书可以帮助 iOS 开发工程师、编程爱好者更深入地理解 iOS 开发原理，更高效地开发出高质量的应用程序。

图书在版编目（CIP）数据

iOS 性能优化实战 / 珲少编著. —北京：电子工业出版社，2019.5
ISBN 978-7-121-36152-4

Ⅰ. ①i… Ⅱ. ①珲… Ⅲ. ①移动终端—应用程序—程序设计 Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字（2019）第 048976 号

责任编辑：高洪霞
印　　刷：北京季蜂印刷有限公司
装　　订：北京季蜂印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　　印张：27　　字数：518.4 千字
版　　次：2019 年 5 月第 1 版
印　　次：2019 年 5 月第 1 次印刷
定　　价：99.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）51260888-819，faq@phei.com.cn。

前　言

首先感谢你购买本书作为提高 iOS 开发技能的学习教程。作为一名最前沿的程序开发者，我非常理解当需要深入学习一门技术却无从下手时的迷茫与急迫感。因此在编写本书时，我尤其注意将重点、难点及开发过程中极易被忽视却十分重要的点突出讲解，希望能够帮助你用最短的学习时间，得到最显著的提高。

阅读本书目录，你会发现，本书不是一本基础的 iOS 开发教程，如果你没有丝毫的 iOS 开发基础，那么本书对你来说可能会有难度而且不易理解，如果真是这样，我建议你先学习 iOS 基础开发知识，再来阅读本书，一定会受益匪浅。

学习一门编程语言、掌握一种程序开发技术会让你从中获得极大的乐趣。科技领域的高速发展使得企业和公司需要越来越多的编程技术人员，市场上的编程书籍琳琅满目，其中的大多数侧重于基础入门与应用实战，就 iOS 开发来说，如果你想深入地了解这门技术，那么本书是非常好的选择。

本书内容及体系结构

本书分为 9 章，内容规划如下。

第 1 章介绍开发一款高性能的 iOS 应用需要关注的性能指标，并介绍了如何使用工具来监测和优化应用程序的性能。之前你可能只是发现某个界面会卡顿，在学习本章内容后你将可以找到具体是由哪一行代码造成的卡顿。

第 2 章介绍 iOS 开发中的内存管理技术，如果你是 iOS 开发初学者，那么你可能从未关心过内存管理的问题，但是随着学习的深入，你需要清楚地了解内存管理的原理和需要注意的事项。在混合开发、底层框架开发中，内存管理更是重中之重。要成为一名卓越的

iOS 开发工程师，这是你必须迈出的一步。

第 3 章介绍网络技术在 iOS 开发中的应用，并提供一些思路来解决网络卡顿问题，提升用户体验。还会介绍一些常用的辅助工具，帮助你对网络进行检查，对当前网络状态进行快照或模拟操作。

第 4 章介绍应用程序的启动流程及与启动流程相关的一些技术点，如推送的应用、高级的 VoIP 推送和网络电话功能的应用。

第 5 章深入介绍 iOS 视图与动画的相关内容，重点介绍 UITableView 组件的优化方法与思路，完整介绍动画技术在 iOS 开发中的应用。

第 6 章系统地介绍 iOS 多线程开发技术的应用，包括 NSThread、NSOperation 及高级的 GCD 相关用法。

第 7 章介绍动态特性与运行时，这部分内容也是 iOS 程序开发的高级技巧，在很多情况下你可能都不需要使用到这些技术，但是学习它们可以帮助你更好地理解程序的运行原理，也可以帮助你一眼看出一些奇怪问题出现的核心原因。

第 8 章介绍 JavaScript 技术在 iOS 开发中的应用，主要介绍 Native 与 WebView 的交互方式以及高级的 JavaScript 线程应用。在学习本章之后，你就能够很轻松地理解 Hybird App 混合开发的原理。

第 9 章是本书的扩展章节，主要介绍 App Extension 的相关应用及数据交互的方法，灵活地使用 App Extension 可以为你的应用程序添加有趣而实用的新功能。

希望本书可以帮助你达成自己的学习目标，下面的博客是我几年来的编程生活积累的一些财富，里面有关于 iOS 开发、Android 开发、前/后端开发以及编程语言相关的 400 余篇博客，如果需要，你可以挑选自己感兴趣的内容阅读：

https://my.oschina.net/u/2340880

你也可以在下面的网站找到我的一些教学资源：

https://edu.csdn.net/lecturer/course_list

我是一名知识传播者，也是学习者，如果你在学习的过程中，遇到任何问题或者发现了本书的遗漏或错误之处，可以与我联系，我的 QQ 号码是：316045346。当然，在出版前，我和编辑以及所有校验和整理本书的老师都付出了很多汗水，尽量保证让它尽善尽美地呈现在你的面前。

最后，感谢编辑在本书编写过程中提出的宝贵意见和在修订过程中的辛苦工作，感谢

吕远、练向、帅坤、东科以及其他同学和同事在生活和工作中给我的帮助和启发，大家经常性的技术讨论使我受益颇多。感谢其他所有为本书出版付出汗水的人们。如果本书可以给你带来提高与帮助，那么这一切都是值得的。

本书读者对象

- 在职的 iOS 开发工程师
- 编程爱好者
- iOS 开发初学者
- 关注 iOS 项目优化的工程师
- 需要深入学习 iOS 程序工作原理的工程师

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- 下载资源：本书提供示例代码及资源文件，均可在 下载资源 处下载。
- 提交勘误：您对书中内容的修改意见可在 提交勘误 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- 交流互动：在页面下方 读者评论 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：*http://www.broadview.com.cn/36152*

目　录

第 1 章　关于性能你需要知道的事 …… 1

1.1　衡量应用程序性能优劣的一些标准 …… 1

1.1.1　代码的执行效率 …… 2

1.1.2　内存占用 …… 4

1.1.3　CPU 负担与能耗 …… 5

1.1.4　动画流畅度 …… 7

1.1.5　网络缓存 …… 8

1.1.6　应用程序启动时间 …… 9

1.1.7　应用程序包尺寸 …… 9

1.2　Xcode 断点与静态分析工具 …… 10

1.2.1　添加自定义断点 …… 10

1.2.2　为自定义断点添加行为 …… 11

1.2.3　添加全局类型的断点 …… 13

1.2.4　Xcode 的静态分析工具 …… 14

1.3　Instruments：性能分析和测试工具 …… 16

1.3.1　Activity Monitor：活动监视器 …… 16

1.3.2　Allocations：内存跟踪工具 …… 17

1.3.3　CoreAnimation：核心动画监测工具 …… 19

1.3.4　Counters：仪表计数器 …… 20

1.3.5 Energy Log: 能耗记录器 ····· 21
1.3.6 Leaks: 内存泄漏检查工具 ····· 22
1.3.7 Network: 网络连接检查工具 ····· 23
1.3.8 自定义 Instruments 工具模板 ····· 23
1.4 使用 LLDB 调试工具 ····· 25
1.4.1 使用 expression 指令进行动态代码执行 ····· 25
1.4.2 使用 frame 指令查看代码帧信息 ····· 27
1.4.3 使用 thread 相关指令操作线程 ····· 29
1.4.4 其他 LLDB 常用指令 ····· 33
1.5 日志与埋点 ····· 34
1.5.1 异常分析 ····· 35
1.5.2 使用 Bugly 异常捕获工具 ····· 41
1.5.3 应用程序埋点 ····· 43
1.5.4 使用 Fabric 分析工具 ····· 43
第 2 章 iOS 内存管理 ····· 45
2.1 iOS 的内存管理模型 ····· 45
2.1.1 关于内存消耗与引用计数 ····· 45
2.1.2 MRC 内存管理 ····· 46
2.1.3 关于 ARC ····· 49
2.1.4 属性修饰符 ····· 51
2.1.5 ARC 与 MRC 进行混编 ····· 53
2.2 自动释放内存 ····· 54
2.2.1 关于 autorelease 方法 ····· 55
2.2.2 自动释放池 ····· 57
2.2.3 系统维护的自动释放池 ····· 59
2.3 杜绝内存泄漏 ····· 60
2.3.1 Block 与循环引用 ····· 61
2.3.2 代理与循环引用 ····· 63

2.3.3 定时器引起的内存泄漏 ····· 65
2.4 关于“僵尸”对象 ····· 66
2.4.1 捕获“僵尸”对象 ····· 66
2.4.2 处理“僵尸”对象 ····· 68
2.5 CoreFoundation 框架中的内存管理 ····· 71
2.5.1 CoreFoundation 中的引用计数 ····· 71
2.5.2 CoreFoundation 框架与 Foundation 框架混用 ····· 72
2.6 扩展：关于 id 与 void* ····· 74
2.6.1 关于 id 类型 ····· 74
2.6.2 关于 void 与 void* ····· 75
2.6.3 解决最初的问题 ····· 76
第 3 章 应用程序网络与能耗优化 ····· 77
3.1 深入 iOS 网络开发技术 ····· 77
3.1.1 初识 NSURLSession ····· 79
3.1.2 NSURLConnection 的简单应用 ····· 83
3.1.3 请求对象 NSURLRequest ····· 85
3.1.4 请求回执对象 NSURLResponse ····· 87
3.1.5 数据缓存对象 NSURLCache ····· 88
3.1.6 本地用户凭证对象 NSHTTPCookie ····· 90
3.1.7 使用第三方网络诊断库——LDNetDiagnoService_IOS ····· 92
3.2 iOS 网络开发及优化秘技 ····· 96
3.2.1 使用 Charles 抓包工具 ····· 96
3.2.2 使用 Charles 进行 HTTPS 抓包 ····· 100
3.2.3 使用 Charles 进行网络环境模拟 ····· 102
3.2.4 使用 Charles 添加请求断点 ····· 104
3.2.5 使用 Charles 进行数据模拟 ····· 106
3.2.6 Charles 的请求重写功能 ····· 108
3.2.7 使用 Mock.js 搭建本地数据模拟服务 ····· 110

3.2.8 学习使用 JSONModel 库 …… 116
3.2.9 属性自动生成工具 …… 121
3.3 定位与地图 …… 130
3.3.1 使用定位服务 …… 130
3.3.2 原生地图开发 …… 133
3.3.3 添加大头针与自定义标注 …… 136
3.3.4 添加地图覆盖物 …… 138
3.3.5 检索附近兴趣点和导航服务 …… 140
3.4 定时器应用 …… 146
3.4.1 NSTimer 的简单应用 …… 146
3.4.2 关于 RunLoop 的一些探究 …… 148
3.4.3 中心化管理 NSTimer 定时器 …… 149
3.4.4 CADisplayLink 类的应用 …… 153
3.4.5 使用 GCD 方式的定时器 …… 154
第 4 章 从应用程序启动说起 …… 156
4.1 应用程序的启动原理 …… 156
4.1.1 深入 UIApplication 类 …… 156
4.1.2 UIApplication 相关类别介绍 …… 159
4.1.3 关于 UIApplicationDelegate …… 162
4.2 本地通知与远程推送 …… 166
4.2.1 使用本地推送 …… 166
4.2.2 远程推送基础 …… 169
4.2.3 深入理解 UserNotification 框架 …… 174
4.3 PushKit 框架与 CallKit 框架 …… 189
4.3.1 VoIP 与 PushKit …… 190
4.3.2 学习使用 CallKit 框架 …… 192
4.3.3 来电拦截与号码识别 …… 199

第 5 章　深入 iOS 视图与动画 …… 204
5.1　关于视图控制器 …… 204
5.1.1　UIViewController 的生命周期 …… 204
5.1.2　从 StoryBoard 加载 UIViewController 对象的传值陷阱 …… 207
5.1.3　关于 UIViewController 的切换 …… 209
5.2　视图控制器的转场动画 …… 210
5.2.1　UIViewController 的模态跳转转场 …… 211
5.2.2　导航转场动画的自定义 …… 218
5.2.3　UITabBarController 的转场动画 …… 221
5.3　列表视图的性能优化 …… 221
5.3.1　UITableView 的构建原理 …… 221
5.3.2　对 UITableView 可变行高的优化方式 …… 224
5.3.3　关于高度不定的列表分区头、尾视图 …… 228
5.4　iOS 图像绘制技术 …… 230
5.4.1　CGPath 路径类 …… 231
5.4.2　理解图形上下文 …… 238
5.4.3　颜色与色彩空间 …… 252
5.4.4　图形变换函数 …… 258
5.4.5　Patterns 模型的应用 …… 259
5.4.6　绘制梯度渐变视图 …… 263
5.4.7　进行图像处理 …… 268
5.4.8　关于层聚合 …… 272
5.5　iOS 核心动画技术 …… 275
5.5.1　初识 CoreAnimation …… 275
5.5.2　深入理解 CALayer …… 277
5.5.3　几种常用的 CALayer 子类 …… 282
5.5.4　CoreAnimation 动画 …… 287

第 6 章 iOS 多线程开发技术 …… 293

6.1 使用 NSThread 进行线程管理 …… 293

6.1.1 NSThread 中常用类方法 …… 293

6.1.2 NSThread 成员方法和属性的应用 …… 295

6.1.3 隐式地使用 NSThread 进行多线程编程 …… 296

6.2 NSOperation 与 NSOperationQueue 的应用 …… 296

6.2.1 关于 Operation 基类的解析 …… 296

6.2.2 NSBlockOperation 类的应用 …… 297

6.2.3 NSInvocationOperation 类的应用 …… 299

6.2.4 操作之间的依赖关系 …… 300

6.2.5 NSOperationQueue 操作队列的应用 …… 301

6.3 学习使用 GCD …… 303

6.3.1 GCD 的调度机制 …… 303

6.3.2 添加任务到调度队列中 …… 304

6.3.3 使用队列组 …… 305

6.3.4 GCD 对循环任务的处理 …… 308

6.3.5 GCD 中的消息与信号 …… 309

6.3.6 队列的挂起与开启 …… 310

6.3.7 使用 GCD 处理延时任务 …… 310

6.3.8 数据存取的线程安全问题 …… 311

6.3.9 GCD 模式的单例 …… 313

6.3.10 关于 GCD 中的内存管理 …… 314

第 7 章 iOS 运行时技术 …… 315

7.1 动态的 Objective-C 语言 …… 315

7.1.1 窥探消息转发机制 …… 316

7.1.2 消息传递与继承链 …… 318

7.1.3 拯救未知消息的三根救命稻草 …… 320

7.1.4 你真的需要救命稻草吗 …… 324
7.1.5 发送消息相关的几个函数 …… 328
7.2 运行时方法解析 …… 331
7.2.1 与运行时相关的类操作函数 …… 332
7.2.2 与运行时相关的实例对象属性操作函数 …… 334
7.2.3 与运行时相关的实例对象方法操作函数 …… 337
7.2.4 与运行时相关的协议操作函数 …… 340
7.3 运行时特性的基本应用 …… 341
7.3.1 操作变量的巧妙方法 …… 341
7.3.2 操作方法的巧妙方法 …… 345
7.4 使用运行时动态修改 UILabel 的默认字体 …… 348
7.4.1 使用框架统一处理 …… 348
7.4.2 使用运行时函数替换 UILabel 的初始化方法 …… 348
7.5 设置可自动归档的数据模型基类 …… 352
第 8 章 JavaScript 与 Native 交互技术的应用 …… 354
8.1 JavaScriptCore 框架详解 …… 354
8.1.1 JavaScriptCore 框架中的几个核心类 …… 355
8.1.2 在 Native 中运行 JavaScript 脚本代码 …… 356
8.1.3 在 JavaScript 脚本中调用 Objective-C Native 方法 …… 358
8.1.4 深入 JSContext 类 …… 359
8.1.5 深入 JSValue 类 …… 360
8.1.6 Objective-C 与 JavaScript 复杂对象的映射 …… 365
8.1.7 C 语言风格的 API …… 366
8.1.8 设计 Hybird App 框架 …… 371
8.2 WebKit 框架的应用 …… 377
8.2.1 WebKit 框架概览 …… 377
8.2.2 使用 WKWebViewConfiguration 对 WebView 进行配置 …… 379
8.2.3 WKWebView 中的属性和方法解析 …… 382

8.2.4 WKWebView 中的 JavaScript 与 Native 交互 …… 384
8.2.5 WKNavigationDelegate 协议中的方法解析 …… 385
8.2.6 WKUIDelegate 协议中的方法解析 …… 388

第 9 章 iOS 扩展开发 …… 390

9.1 Today 扩展的应用 …… 390
9.1.1 创建 Today 扩展程序 …… 391
9.1.2 Today 扩展与宿主应用程序进行数据交互 …… 393
9.2 分享扩展的应用 …… 396
9.3 照片编辑扩展 …… 399
9.4 自定义键盘扩展 …… 400
9.4.1 了解 UIInputViewController 类 …… 400
9.4.2 创建自定义的数字输入键盘 …… 401
9.5 iMessage 扩展的应用 …… 404
9.5.1 开发独立的表情包 …… 404
9.5.2 开发寄宿于宿主应用程序的表情包扩展 …… 406
9.5.3 开发 iMessage App …… 407
9.5.4 对开发 iMessage App 的几点建议 …… 418

第 1 章　关于性能你需要知道的事

应用程序的性能优劣，可能并没有引起很多开发者甚至软件公司足够的重视。毕竟一款应用程序的“功能”决定用户是否需要用，“界面”决定用户是否喜欢用，并且，在一款应用程序没有实现需求前，过早地纠结性能会浪费大量的时间和人力成本。然而尽管如此，性能却是在一款优秀产品迭代过程中必须考虑的事情。

所谓性能，无非是一种指标，在软件开发中，该指标往往会关注两个方面：效率和消耗。效率主要是指代码的执行效率、动画的流畅度、应用的冷启动时间和热启动时间、网络通信的阻塞时间，等等。消耗主要是指内存的消耗、有没有内存泄漏、CPU 的占用率、耗电与应用程序包尺寸，等等。

本章主要讨论在 iOS 应用程序的性能优化中，有哪些性能指标是用户需要考虑的，并将介绍如何使用工具来分析一款 iOS 应用程序的性能问题。最后，还将讨论如何监控和分析已发布产品的行为。

1.1　衡量应用程序性能优劣的一些标准

一款应用程序的性能优劣是由多项标准来衡量的。所有用户体验优秀的应用程序在被开发时无疑都会被关注其在运行过程中的性能情况。对于用户而言，最为直观的体验就是运行速度，如果应用程序的启动时间超过 5 秒，那么 30%以上的用户都会选择关掉它。

如果一款应用程序启动时连续闪退超过 3 次，那么半数以上的用户会选择卸载。应用界面加载时间超过 3 秒或动画有卡顿也会给用户带来不适感。

1.1.1 代码的执行效率

随着硬件设备越来越强，在开发中，我们往往会忽略具体某段代码的执行效率。然而当项目足够复杂、数据处理量足够大时，代码执行效率往往是最难优化的。

在编程中，一段代码的执行效率实际上很难被估算和预测。其主要受如下几个方面的影响：

- 算法依据的数据基础。
- 编译器产生的代码质量和语言的执行效率。
- 问题的输入规模。
- 硬件的执行速度。

在通常情况下，问题的输入规模和算法的数学基础是开发者需要考虑的因素。“时间复杂度”是用来描述算法执行效率的一个重要标准。

在理解时间复杂度之前，你应该先了解什么叫作算法的时间频度。所谓时间频度，即一个算法解决问题所消耗的时间。但是在一般情况下，一个算法解决问题消耗的时间通常与输入的值有关，例如我们输入一个整数，要找到比它小的所有正偶数，则示例代码如下：

```
int main(int argc, const char * argv[]) {
    @autoreleasepool {
        int n = 10;
        for (int i=0; i<n; i++) {
            if (i%2==0) {
                printf("%d\n", i);
            }
        }
    }
    return 0;
}
```

当输入 n 的值为 10 时，该循环会执行 10 次，如果设此时间频度为 t，则当输入 n 的值为 20 时，时间频度相应为 2t。时间复杂度用来描述随着问题规模 n 的变化，时间频度 t 的变化规律。

注意：在一般情况下，算法中基本操作重复执行的次数是问题规模 n 的某个函数，用 T(n)表示，若有某个辅助函数 f(n)，使得当 n 趋近于无穷大时，T(n)/f(n)的极限值为不等于